An AGI architecture in vector space

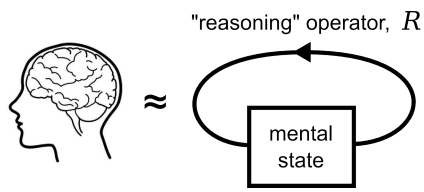
甄景贤 (King-Yin Yan) and Juan Carlos Kuri Pinto

General.Intelligence@Gmail.com

Abstract. This is a draft.

1 Main idea

The main idea is to regard “thinking” as a **dynamical system** operating on **mental states**:



(1)

For example, a mental state could be the following set of propositions:

- I am in my room, writing a paper for AGI-16.
- I am in the midst of writing the first sentence, “The main idea is...”
- I am about to write an infinitive phrase “to regard...”

Thinking is the process of **transitioning** from one mental state to another.

By representing a cognitive state as a vector $\boldsymbol{x} \in X$ where X is the cognitive state-space, the reasoning operator R as a map $X \to X$, we would have at disposal all the tools available in vector space such as:

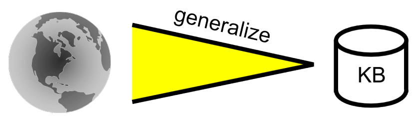
- numerical optimization (including gradient descent)
- differential equations governing time evolution
- dynamical systems theory, control theory (eg. adaptive filters)
- Lie algebra and C^*-algebra of continuous operators
- matrix theory, iteration and fixed-point theory
- dynamic programming (aka. reinforcement learning)
- neural networks and deep learning ... etc.

This is also partly inspired by the success of Google’s PageRank and Word2Vec algorithms.

2 Relation to Logic-based AI (LBAI)

In this paper we would jump back and forth between the logic-based view and the dynamical state-space view.

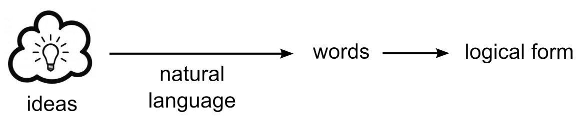
LBAI can be viewed as the compression of a world model into a knowledge-base (KB) of logic formulas:



(2)

The world model is *generated* combinatorially from the set of logic formulas, vaguely reminiscent of a "basis" in vector space. The generative process in logic is much more complicated, contributing to its high *compressive* ability on the one hand, and the *complexity* of learning such formulas on the other hand.

In LBAI the knowledge representation structure is built (*fixed*) from the bottom up:



(3)

but is it valid (or profitable) to assume that our mental representations are *isomorphic* to such logical structures? Or drastically different?

Humans are good at designing symbolic structures, but we don't know how to design *neural* representations which are more or less opaque to us. Perhaps we could use a neural network acting recurrently on the state vector to **induce** an internal representation of mental space. "*Induced by what*," you ask? By the very structure of the neural network itself. In other words, forcing a neural network to *approximate* the ideal operator R^*.

From an abstract point of view, we require:

- R be an endomorphism: $X \to X$
- R has a learning algorithm: $R \overset{A}{\longmapsto} R^*$

R would contain all the knowledge of the KB, so we expect it to be "large" (eg. having a huge number of parameters). We also desire R to possess a **hierarchical** structure because hierarchies are computationally very efficient. A multi-layer perceptron (MLP) seems to be a good candidate, as it is just a bunch of numbers (weight matrices W) interleaved by non-linear activation functions:

$$R(\boldsymbol{x}) = \sigma(W_1 \sigma(W_2 ... \sigma(W_L \boldsymbol{x}))) \tag{4}$$

where L is the number of layers. MLPs would be our starting point to explore more design options.

The idea of R as an operator acting on the state is inspired by the "consequence operator" in logic, usually denoted as Cn:

$$\mathrm{Cn}(\mathit{\Gamma}) = \{ \text{ set of propositions that entails from } \mathit{\Gamma} \ \} \tag{5}$$

but the function of R can be broader than logical entailment. We could use R to perform the following functions which are central to LBAI:

– **deduction** – forward- and backward-chaining
– **abduction** – finding explanations
– **inductive learning**

Example: primary-school arithmetic

A recurrent neural network is a *much more powerful* learning machine than a feed-forward network, even if they look the same superficially.

$$\begin{array}{rcc} & 7 & 3 \\ - & 3 & 7 \\ \hline & 3 & 6 \end{array}$$

As an example, consider the way we perform 2-digit subtraction in primary school. This is done in two steps, and we put a dot on paper to mark "carry-over".
The use of the paper is analogous to the "tape" in a Turing machine – the ability to use short-term memory allows us to perform much more complex mental tasks.
We did a simple experiment to train a neural network to perform primary-school subtraction. The operator is learned easily if we train the 2 steps *separately*. The challenge is to find an algorithm that can learn **multi-step** operations by itself.

In LBAI, logic possesses additional structure:

– **truth values** (eg. P(rain tomorrow) = 0.7)
– **propositional structure** (eg. conjunction: $A \wedge B$)
– **sub-propositional structure** (eg. predication: loves(john, mary))
– **subsumption structure** (eg. dog $\subseteq$ animal)

These structures can be "transplanted" to the vector space X via:

– **truth values:** an extra dimension conveying the "strength" of states
– **propositional structure:** eg. conjunction as vector addition,

$$A \wedge B = \boldsymbol{x}_A + \boldsymbol{x}_B + ... \tag{6}$$

but we have to avoid linear dependencies ("clashing") such as:

$$\boldsymbol{x}_3 = a_1 \boldsymbol{x}_1 + a_2 \boldsymbol{x}_2 \tag{7}$$

This would force the vector space dimension to become very high.
– **sub-propositional structure:** eg. tensor products as composition of concept atoms:

$$\text{loves(john, pete)} = \overrightarrow{john} \otimes \overrightarrow{love} \otimes \overrightarrow{pete} \tag{8}$$

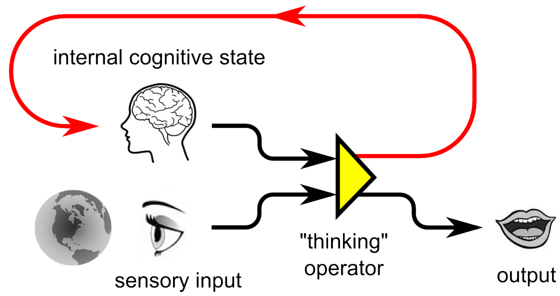
– **subsumption structure:** eg. define the positive cone C such that

$$\text{animal} \supseteq \text{dog} \quad \Leftrightarrow \quad \overrightarrow{animal} - \overrightarrow{dog} \in C \tag{9}$$

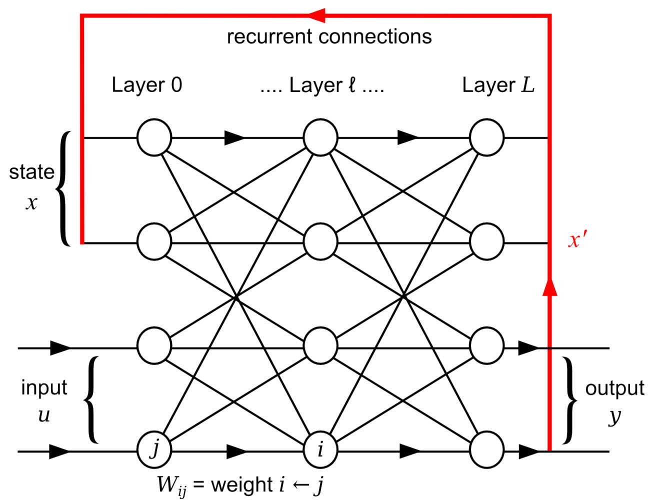
But the more logical structure we add to X, the more it will resemble logic, and this whole exercise becomes pointless. Remember our original goal is to try something different from logic, by *relaxing* what defines a logical structure. So we would selectively add features to X.

3 Architecture

First, cartoon version:

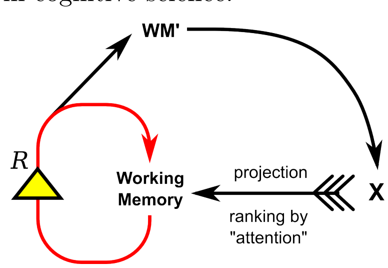


(10)



(11)

TO-DO: The state space X may be too large and we may need an **attention mechanism** to select some parts of X for processing by R. This is the notion of **working memory** in cognitive science.



(12)

4 Deep Recurrent Learning

The learning algorithm for R is central to our system. R learns to recognize input-output pairs $(\boldsymbol{x}_0, \boldsymbol{x}^*)$. What makes it special is that R is allowed to iterate a *flexible* number of times before outputting an answer. In feed-forward learning we simply learn single-pass recognition, whereas in common recurrent learning we train against a *fixed* time sequence. Here, the time delay between input and output is allowed to stretch arbitrarily.

Suppose the recurrent network R iterates n times:

$$\boldsymbol{x}' = \overbrace{R \circ R \circ ...}^{n}(\boldsymbol{x}) \tag{13}$$

As $n \to \infty$, we get the continuous-time version (a differential equation):

$$\frac{d\boldsymbol{x}(t)}{dt} = R(\boldsymbol{x}(t)) \tag{14}$$

We could run the network R for a long enough time T such that it is highly likely to reach an equilibrium point. Then:

$$\boldsymbol{x}' = \int_0^T R(\boldsymbol{x}(t))dt \tag{15}$$

and the error:

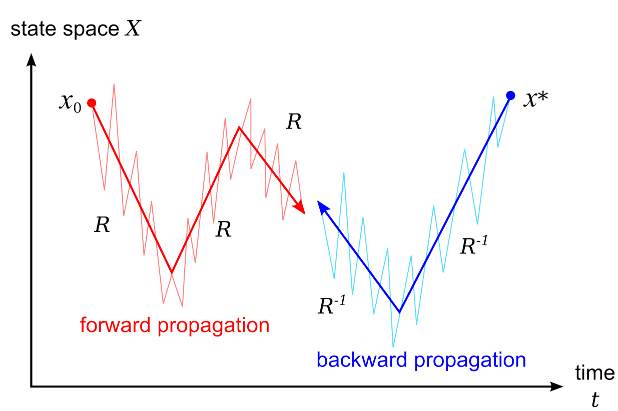
$$\mathscr{E} = \boldsymbol{x}^* - \boldsymbol{x}' \tag{16}$$

where $\boldsymbol{x}^*$ is the target value which is independent of time.

$$\begin{aligned}\frac{\partial \mathscr{E}}{\partial \boldsymbol{W}} &= -\frac{\partial}{\partial \boldsymbol{W}} \int_0^T R(\boldsymbol{x}(t))dt \\ &= -\frac{\partial}{\partial \boldsymbol{W}} \int_0^T \sigma(W_1 \sigma(W_2 ... \sigma(W_L \boldsymbol{x}(t)))dt\end{aligned} \tag{17}$$

4.1 Forward-backward Algorithm

This is inspired by forward- and backward-chaining in LBAI. We propagate the state vector from both the initial state $\boldsymbol{x}_0$ as well as the final state $\boldsymbol{x}^*$. This bi-directional propagation is added with noise and repeated many times, thus implementing a **stochastic local search**:



(18)

When the forward and backward states get close enough, a successful path is found, and we record the gap and the noises along the path, and use them to train R so that this new path would be recognized.

Acknowledgements

In a forum discussion with Ben Goertzel dated 25 June 2014 on the AGI mailing-list: (artificial-general-intelligence @googlegroups.com), YKY asked: Why bother with neural networks, which typically require many neurons to encode data, when logic-based AI can represent a proposition with just a few symbols? Ben's insight is that neural networks are capable of learning its own representations, and their learning algorithms are relatively fast. We have been working on "neo-classical" logic-based AI for a long time, and begin to realize that inductive learning in logic (based on combinatorial search in a symbolic space) is perhaps *the bottleneck* in the entire logic-based paradigm. So we try to look for alternatives that might enable learning to be faster, though we would still emphasize that logic-based AI remains a viable approach to AGI.